Through the Eyes of a Programmer

AK WI

Herausgeber

Martin R. Wolf
Thomas Barton
Frank Herrmann
Vera G. Meister
Christian Müller
Christian Seel

Angewandte Forschung in der Wirtschaftsinformatik 2019

unterstützt durch:

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Arbeitskreis Wirtschaftsinformatik an Hochschulen für Angewandte Wissenschaften im deutschsprachigen Raum (AKWI)

Angewandte Forschung in der Wirtschaftsinformatik 2019

Tagungsband zur 32. AKWI-Jahrestagung
vom 15.09.2019 bis 18.09.2019 an der
Fachhochschule für Angewandte Wissenschaften Aachen

herausgegeben von
Martin R. Wolf, Thomas Barton, Frank Herrmann, Vera G. Meister,
Christian Müller, Christian Seel

unterstützt durch
Fachhochschule für Angewandte Wissenschaften Aachen

mana-Buch, Heide

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie, detaillierte bibliografische Daten sind im Internet über www.dnb.de abrufbar.

Angewandte Forschung in der Wirtschaftsinformatik 2019
Tagungsband zur wissenschaftlichen Fachtagung am 16.09.2019 anlässlich der 32. Jahrestagung des Arbeitskreises Wirtschaftsinformatik an Hochschulen für Angewandte Wissenschaften im deutschsprachigen Raum (AKWI) vom 15.09.2019 bis 18.09.2019 an der Hochschule für Angewandte Wissenschaften Aachen

Herausgeber:
Martin R. Wolf, Hochschule für Angewandte Wissenschaften Aachen, m.wolf@fh-aachen.de
Thomas Barton, Hochschule Worms, barton@hs-worms.de
Frank Herrmann, Ostbayerische Technische Hochschule Regensburg, frank.herrmann@oth-regensburg.de
Vera G. Meister, Technische Hochschule Brandenburg, vera.meister@th-brandenburg.de
Christian Müller, Technische Hochschule Wildau [FH], christian.mueller@th-wildau.de
Christian Seel, Hochschule für Angewandte Wissenschaften Landshut, christian.seel@haw-landshut.de

Redaktionsschluss: 26.06.2019
Erscheinungstermin: 16.09.2019

Die Herstellung dieses Tagungsbandes erfolgte mit freundlicher Unterstützung durch:
Hochschule für Angewandte Wissenschaften Aachen

Verlag: mana-Buch, Feldblick 24, 25746 Heide, Germany, www.mana-Buch.de
Druck: Amazon Fulfillment
ISBN: 978-3-944330-62-4

Through the Eyes of a Programmer: A Research Project on how to Foster Programming Education with Eye-Tracking Technology

Selina Emhardt, Christian Drumm, Tamara van Gog, Saskia Brand-Gruwel, Halszka Jarodzka

Abstract

Nowadays, there is a high demand for programming expertise on the labor market. New technologies such as eye tracking could help to improve programming education and thereby help to fulfill this demand. For instance, Eye Movement Modeling Examples (EMMEs) are learning videos that visualize a person's (the model's) eye movements while s/he demonstrates how to perform a (programming) task. The eye movements can, for instance, get visualized as moving dots onto a screen recording. By observing where an expert programmer looks, programming beginners might better understand what s/he is doing and referring to. Recent studies showed promising first results about the beneficial effects of using EMMEs in programming education. In this manuscript, we present a research project that aims to provide evidence-based guidelines for educational practitioners on how to use eye-tracking technology for programming training. We first introduce the basic concept of EMMEs and exemplary gaps in literature. We then present our first empirical study on how different instructions affect expert programmer's eye movements when modeling a debugging task (and hence EMME displays). With this manuscript, we hope to inspire more programmers to use eye-tracking technology for programming education.

1 Introduction

A recent bitkom study (see [PaBa18]) showed an increase of open IT positions in Germany in 2018 by 49%. Especially software developers are nowadays in high demand on the labor market and an effective programming education is needed to fulfill this demand. New learning technologies are gradually becoming an integral part of programming education. For instance, screencasts (desktop video recordings usually accompanied by the presenter's video and narration) are a promising tool for programming education (see [KeSt18]). But what information should learning videos ideally display and how can we optimize their design?

From educational research we know that observing a 'model' (e.g., an expert, teacher, or more competent peer student) who demonstrates how to perform a task successfully can be a powerful way of acquiring new knowledge and skills and has often found to be more effective and efficient than learning by problem-solving alone (e.g., see [GoRu10]). Implementing this principle into educational settings is referred to as 'modeling' (see e.g., [Renk14], [GoRu10]).

For many tasks such as programming, not all aspects of performance can simply be observed from outside. Instead, the model needs to 'externalize' his or her thoughts by telling (verbalizing) what he or she is doing to perform this task. Often, however, a verbal explanation is not sufficient. For instance, when the learner does not know what exactly the model is

referring to in his or her explanation, the learner is not able to follow the explanation. In such cases, Eye-Movement Modeling Examples (EMME) have the potential to foster learning. EMME are screencast videos that show a model performing the task, often including an audio of his or her verbal explanation, plus a visualization of the focus of visual attention of the model (e.g., see [Go++09]). Like that, the learner sees on the video which element on the screen the model was looking at while explaining the task (see Figure 2 below). To create EMMEs for programming education, an expert model solves a programming task on the computer while his/her activities on the screen as well as the corresponding verbal explanations are recorded (screen and audio recording). An eye tracking device, which is attached to the computer screen (Figure 1), captures the eye movement of the model (i.e., where the model looks on the screen) over the course of time (for a general introduction to eye-tracking see [Ho++11]). Afterwards, the EMME are generated as an integrated video of the screen recording with the voiceover and a visualization of the eye movements. The eye movements can be visualized in the EMME in different ways, for instance, fixations can be shown as a moving dot, a circle or a spotlight (e.g., [Ja++], [Ja++13]). A fixation refers to the time the eye is relatively still for about 200-400 milliseconds. During a fixation, the eye takes in information. In an EMME, the visualization of the model's eye movements rests still on the location of interest during a fixation. Saccades are the fast eye movements between fixations, which relocate the focus of attention, which can be inferred from the EMME.

By observing the model's eye movements, learners can better understand what the model is referring to and what the model is doing. In this way, EMME videos were effective to guide novices' attention and foster learning in various domains, such as medicine or biology (see e.g., [Ja++], [Ja++13]). A recent study showed promising first results that displaying a programmer's eye-movements can foster learners' understanding and performance in source code comprehension (see [Be++18]). Despite these promising findings, we know from prior research in other domains that EMME are not always effective (e.g., see [Go++09], [MWJG16]). Therefore, the conditions under which EMME foster learning need to be further investigated.

Hence, in our research project, we aim at systematically investigating various influencing factors of learning with EMME (e.g., characteristics of the model, visualization of eye movements). This can provide evidence-based guidelines for educational practitioners on how to design EMMEs for their programming training. In this manuscript, we first explain the basic concept of EMMEs and why they are a promising tool to foster learning in the domain of programming education. Then, we will highlight exemplary gaps in EMME literature, such as the right choice of model instruction. Next, we introduce our first empirical study about how different model instructions affect EMME displays and finally present future studies that could tackle subsequent open questions. We hope to inspire more programmers to consider and use eye-tracking technology for programming education.

2 Why EMMEs could foster programming learning

It is already well-established that the level of experience influences eye movements in various ways (e.g., see [ReSh11]). For instance, beginners in a task often attend to less relevant information than highly experienced performers. Eye-tracking studies from programming research also show that the attention focus (i.e., where programmers look) on different

code fragment can set apart less and more experienced programmers (see [AsCr06]). For instance, more experienced programmers attend more to complex statements (see [CrSW02]) but less to code comments (also [CrSt90]).
Such differences in what beginning and expert programmers look at might hamper their communication, because they are not attending to the same objects at the same time. This is especially problematic in situations in which a lot of visual information is presented at once (e.g., a large function with complex statements during debugging). In such situations, even an expert's verbal explanations might not be sufficient to guide novices' attention to relevant areas. For instance, programming experts might use abstract names of functions and symbols that the beginners have not learned yet (e.g., referring to "commenting lines" when talking about the symbol of '#'). Furthermore, (programming) experts might even lose conscious control over some automatized processes (see [SaFl97]), which should make it difficult to verbalize and teach them to beginners.
This is especially problematic in video-based education, since learners cannot ask for clarification when they need it.
One main idea behind EMMEs is that displaying an expert's eye-movements reveal the model's thought processes and guides the learners' attention to the relevant objects. This can clarify what the model is referring to. For instance, when an expert programmer mentions the "commented line" in an EMME, the learner could additionally see that the programmer looks at the information behind the #-symbol. Sharing eye-movement information could furthermore foster a communication state in which the communication partners (e.g., model and learners) look at the same object at the same time (also called joint attention, e.g., [Butt95]).
For the domain of programming research, [StBr04] found that observing another programmer's eye movements during debugging indeed increases the later debugging speed of the observers. More recently, [Be++18] investigated if EMME-based interventions can improve reading and comprehending skills for source code reading. In this experiment, participants' comprehension strategies were taught in an educational EMME video. Participants followed the eye-movements of an advanced programmer performing the task naturally (i.e., as usual and without any specific instructions). A voice-over commented the programmer's strategy in a didactical manner, hence explaining how ideal viewing behavior for source code comprehension should be performed (based on the block model by [Schu08]). The authors concluded that this EMME intervention later resulted in improved task performances of the learners. While this study indicates that EMMEs are a promising tool for programming education, there are still various unanswered questions about the ideal way to design EMMEs.

3 First step in our project: Investigating influences of model instruction on EMME displays

When comparing EMME videos from various studies, it is striking that the instructions that were given to the models to create the videos differ widely. Sometimes, models were instructed to explain their performance in a way that is specifically targeted to beginners. In these cases, the models were instructed to behave in a 'didactic' manner by keeping the knowledge state of the inexperienced audience in mind (e.g., in [Ja++] and [Ja++13]). In

various other studies, however, the models did not receive explicit didactic instructions and hence behaved more naturally (e.g., see [Li++10], [StBr04]).
The aim of our first project study was to explore how the instruction to behave didactically alters expert programmers' displayed behavior (i.e., eye-movements and mouse clicks during code debugging). We recruited 23 professional programmers (experts) and collected data directly at their work offices with a mobile eye-tracking device (see Figure 1, 250Hz SMI RED250 infrared remote mobile eye tracker from SensoMotoric Instruments GmbH, Teltow, Germany). As can be seen in Figure 1, this device is small and easy to transport. It can simply be attached to a laptop, which allowed us to test the participants individually at various locations.

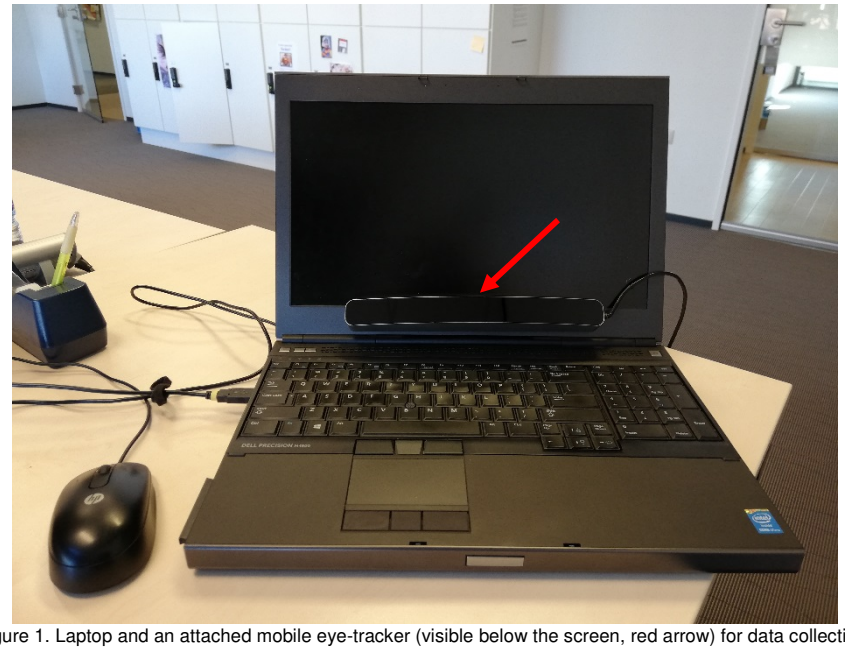

Figure 1. Laptop and an attached mobile eye-tracker (visible below the screen, red arrow) for data collection.

The expert programmers debugged three short Python code snippets. First, they debugged all code snippets 'naturally' (i.e., without a specific instruction) in an interactive IDE (Spyder, https://www.spyder-ide.org/) while we recorded their eye movements, what they said and their mouse clicks. After solving each debugging task, these programming experts were instructed to explain the task solution in a didactical manner to create didactical EMMEs (similar to instructions in previous EMME studies, e.g., [Ja++]). They were instructed to keep the knowledge of a beginning programmer in mind (cf. [JuSB07]). In this way, we recorded experts' didactic verbal explanations, eye movements, and mouse clicks (code running behavior). Figure 2 displays a static visualization of an expert's eye movements. The circles indicate the fixations and the connection lines show where the eye performed fast movements between fixations (saccades). These connecting lines are not present in a EMME video, but the changes in location of the expert's gaze over the course of time are visualized by the movement of the spotlight, circle or dot (i.e., the change in fixation location).
When comparing experts' natural and didactic debugging behavior, our results showed that experts' (viewing) behavior substantially changed, which in turn influences the characteristics of EMME displays. For instance, in comparison to experts' regular debugging behavior,

didactically behaving experts performed longer fixations, executed the code less often, and looked at the code in a more linear manner (i.e., line-wise code reading). Hence, we showed that a didactic instruction affects programming experts' (viewing) behavior and with it the EMME displays with – until now – unknown effects on learning.

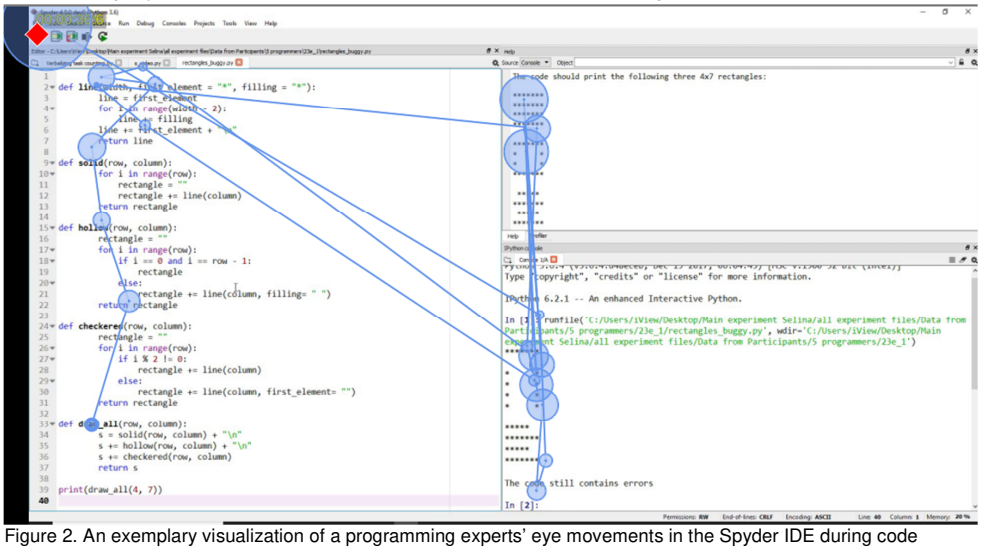

Figure 2. An exemplary visualization of a programming experts' eye movements in the Spyder IDE during code debugging. Fixations are visualized as blue dots and larger dots indicate longer fixations. The lines between the dots indicate fast eye movements between consecutive fixations ('saccades').

4 Open questions, subsequent studies, and related projects

Future research could address how EMMEs could be used as addition to students' regular classes. In practice, students could watch EMMEs for instance as a homework or for self-study, serving a similar purpose as videos on online platforms such as YouTube or Khahn-academy. To use EMMEs effectively for future programming education, we need evidence-based design guidelines, for instance about the most effective model instructions (didactical or natural) to foster learning. In our first study we showed that experts' displayable behavior changes substantially when behaving didactically. While it is clear that these changes affect the characteristics of the EMME displays, we do not yet know how these expert instructions effects learning with EMMEs. As a next step we therefore need to investigate which expert instruction is most suitable to create effective EMMEs. This will be the first study to compare the effect of model instructions on learning with EMMEs, which will provide recommendations on how to instruct models for EMME videos to foster programming education.

Aside from this, there are many more open questions that are worth investigating in future research. For instance, how do the expert models' differences in programming performance affect observers' learning and could displaying several models with large performance differences be even advantageous for learning and knowledge generalization? What are programming teachers' opinion on using EMMEs in their practice? What are concrete difficulties and challenges when implementing EMME in education? Could EMMEs be used as an (online) lecture addition? And can displaying a programming teacher's eye movements support learning in an actual classroom setting?

We finally aim at translating our empirical findings into educational practice in a related project funded by the FH Aachen ("Systematische und nachhaltige Qualitätsentwicklung in Studium und Lehre an der FH Aachen") to enable evidence-based teaching methods in programming. In this project, we improve the existing curriculum based on the 4C-ID model (e.g., see [MeCC02]), a well-established framework on how to design teaching material and training tasks for complex skills, such as programming (see [MeKK03]). One important part of this framework is the use of instructional videos for pre-training beginners at specific tasks. In that, we will compare whether and when our EMME approach results in more efficient learning than regular videos. Also, we will study the potentials of EMME for pre-training can be even enhanced by implementing other 4C-ID principles in subsequent training phases, such as automated feedback based on the actual performance of the students. We hope that answering such questions will show us how to use eye-tracking for programming education in the most effective way.

5 Conclusions

With this manuscript, we showed that eye-tracking technology is an innovative and fruitful tool for programming education. Since EMME research is still in its infancy, evidence-based design guidelines such as the ideal model instruction are until often still missing. The results of our first project study showed how model instructions substantially alter a programmer's displayable behavior (eye movements and mouse clicks). However, while we showed that model instructions affect the characteristics of EMME displays, the effects of model instruction on learning with EMMEs needs future investigation. With this manuscript, we aimed to raise programmers' awareness of the potential of eye tracking for programming education. In the long run, we hope that our research project can shed light on the optimal ways to use eye-tracking technology to foster programming education.

References

[AsCr06] Aschwanden, C. und Crosby, M.: Code scanning patterns in program comprehension. In: Proc. of HICSS (2006).

[Be++18] Bednarik, R., Schulte, C., Budde, L., Heinemann, B., Vrzakova, H.: Eye-movement Modeling Examples in Source Code Comprehension: A Classroom Study. In: Proceedings of the 18th Koli Calling International Conference on Computing Education Research (2018), S. 2.

[Butt95] Butterworth, G.: Origins of mind in perception and action. In: Moore, C. und Dunham, P.J. (Hrsg): Joint attention: Its origins and role in development. NY Psychology Press, 1996, S. 29-40

[CrSW02] Crosby, M.E., Scholtz, J. Wiedenbeck, S.: The roles beacons play in comprehension for novice and expert programmers. In: PPIG (2002), S.5.

[CrSt90] Crosby, M. E. und Stelovsky, J.: How do we read algorithms? A case study. In: Computer 23 (1990), S. 25 - 35.

[Ho++11] Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., Van de Weijer, J.: Eye tracking: A comprehensive guide to methods and measures. Oxford University Press, Oxford, 2011.

[Ja++12] Jarodzka, H., Balslev, T., Holmqvist, K., Nyström, M., Scheiter, K., Gerjets, P., Eika, B.: Conveying clinical reasoning based on visual observation via eye-movement modelling examples. In: Instructional Science 40 (2012), S. 813 – 827.

[Ja++13] Jarodzka, H., van Gog, T., Dorr, M., Scheiter, K., Gerjets, P.: Learning to see: Guiding students' attention via a model's eye movements fosters learning. In: Learning and Instruction 25 (2013), S. 62 - 70.

[JuSB07] Jucks, R., Schulte-Löbbert, P., Bromme, R.: Supporting experts' written knowledge communication through reflective prompts on the use of specialist concepts. In: Zeitschrift für Psychologie/Journal of Psychology (2007), S. 237 - 247.

[KeSt18] Kefalas, P. und Stamatopoulou, I.: Using Screencasts to Enhance Coding Skills: the case of Logic Programming? In: Computer Science & Information Systems 15 (2018), S. 34 - 42.

[Li++10] Litchfield, D., Ball, L. J., Donovan, T., Manning, D. J., Crawford, T.: Viewing another person's eye movements improves identification of pulmonary nodules in chest x-ray inspection. In: Journal of Experimental Psychology: Applied 16 (2010), S. 251 - 62.

[PaBa18] Pauly, B. und Barkei, N.: 82.000 freie Jobs: IT-Fachkräftemangel spitzt sich zu. 2018, https://bitkom.de/Presse/Presseinformation/82000-freie-Jobs-IT-Fachkraeftemangel-spitzt-sich-zu. Abruf am 29.03.2019

[ReSh11] Reingold, E. M. und Sheridan, H.: Eye movements and visual expertise in chess and medicine. In: Liversedge, S.P., Gilchrist, I.D., Everling, S. (Hrsg.): Oxford Handbook on Eye Movements. Oxford University Press, Oxford, 2011, S. 528- 550

[Renk14] Renkl, A.: The Worked Examples Principle in Multimedia Learning. In: Mayer, R. E. (Hrsg.): The Cambridge handbook of multimedia learning. Cambridge University Press, Cambridge, 2014, S. 391 – 412.

[SaFl97] Samuels, S. J. und Flor, R. F.: The importance of automaticity for developing expertise in reading. In: Reading & Writing Quarterly: Overcoming Learning Difficulties 13 (1997), S. 107 - 121.

[Schu08] Schulte, C.: Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In: Proceedings of the Fourth international Workshop on Computing Education Research (2008), S. 149 - 160.

[StBr04] Stein, R. und Brennan, S. E.: Another person's eye gaze as a cue in solving programming problems. In: Proceedings of the 6th international conference on Multimodal interfaces (2004), S. 9 - 15.

[Go++09] van Gog, T., Jarodzka, H., Scheiter, K., Gerjets, P., Paas, F.: Attention guidance during example study via the model's eye movements. In: Computers in Human Behavior 25 (2009), S. 785 - 791.

[GoRu10] van Gog, T. und Rummel, N.: Example-based learning: Integrating cognitive and social-cognitive research perspectives. In Educational Psychology Review 22 (2010), S. 155 - 174.

[MWJG16] van Marlen, T., van Wermeskerken, M., Jarodzka, H., van Gog, T.: Showing a model's eye movements in examples does not improve learning of problem-solving tasks. In: Computers in Human Behavior 65 (2016), S. 448 - 459.

[MeCC02] van Merriënboer, J. J., Clark, R. E., De Croock, M. B.: Blueprints for complex learning: The 4C/ID-model. In: Educational technology research and development 50 (2002), S. 39 - 61.

[MeKK03] Van Merriënboer, J. J., Kirschner, P. A., & Kester, L: Taking the load off a learner's mind: Instructional design for complex learning. In: Educational psychologist 38 (2003), S. 5 – 13.

Acknowledgments
This project is funded by the Netherlands Initiative for Education Research (NRO PROO project 405-17-301)

Kontakt

Selina Emhardt
Open University of the Netherlands
Valkenburgerweg 177, 6419 AT Heerlen
T +49 481 222-222, Selina.emhardt@ou.nl

Dr. Halszka Jarodzka
Open University of the Netherlands
Valkenburgerweg 177, 6419 AT Heerlen
halszka.jarodzka@ou.nl

Prof. Dr. Christian Drumm
FH Aachen
Eupener Str. 70, 52066 Aachen
drumm@fh-aachen.de

Prof. Dr. Saskia Brand-Gruwel
Open University of the Netherland
Valkenburgerweg 177, 6419 AT Heerlen
Saskia.Brand-Gruwel@ou.nl

Prof. Dr. Tamara van Gog
Utrecht University
Heidelberglaan 1, 3584 CS Utrecht
t.vangog@uu.nl